04

Métodos adicionales

Transcripción

[00:00] Hola. ¿Cómo están? Vamos a iniciar nuestra clase 5. En nuestra clase 5 vamos a ver formas de ordenar una lista utilizando la clase collections, utilizando el método sort de ArrayList y también utilizando stream. También vamos a ver una pequeña introducción de lambdas, utilizando tal vez el método filter.

[00:25] Para esto vamos a duplicar esta clase 4, vamos a colocar clase 5. Para que no sea repetido vamos a colocar otros valores, colocar Java, colocar el curso de JavaScript, vamos a colocar un curso de PHP, vamos a colocar un curso de Ruby. Los tiempos vamos a mantenerlos igual y listo.

[00:57] Ahora, utilizando la clase collections, podemos utilizarla de la siguiente forma: sort como estuvimos haciendo anteriormente y escribimos cursos. Después, System.out.println(cursos). ¿Qué sucede? Está dando aquí un error. ¿Por qué? Porque está diciendo que no estoy implementando la clase comparable. ¿Qué sucede?

[01:28] Anteriormente nosotros estábamos utilizando stream. Stream ya implementa comparable. Por eso él estaba funcionando correctamente. Ahora nosotros necesitamos implementar esta clase en nuestra clase curso. Para esto vamos a hacer lo siguiente: implements_Comparable, vamos a digitar aquí curso.

[02:05] Una vez hecho esto, vamos a generar nuestras implementaciones. ¿Qué estamos implementando? El método compareTo. El método stream también

tiene compareTo, pero estaba ejecutando correctamente. Entonces aquí voy a colocar. ¿Qué queremos comparar? Queremos comparar el nombre this.nombre.compareTo y vamos a colocar el o.getNombre. Listo.

[02:44] Una vez hecho esto, nuestra clase 5 va a desaparecer lo que estaba en rojo. Y ahora vamos a poder ejecutar. Y aquí ordenó. Para que se vea mejor el orden, vamos a ponerlo de otra forma. Vamos a hacer que JavaScript esté aquí, que PHP esté aquí, Ruby aquí y Java aquí. Vamos a ver aquí.

[03:24] Primero Java, JavaScript, PHP y Ruby, ordenó correctamente. En el caso que queramos un orden inverso sería collections.reverseOrder. Lo ejecutamos y tenemos Ruby, PHP, JavaScript y Java, tal como está aquí. Pero vamos a verlo ahora de otra forma. Ruby, PHP, JavaScript y Java.

[04:09] ¿Qué sucede si no queremos usar este compareTo? No quiero utilizarlo. Bueno, tenemos dos formas. Primero, podemos utilizar aquí. Si bien lo utilizamos, podemos utilizar aquí nuestra lista de cursos.sort, implementamos aquí el comparator, por ejemplo podemos poner aquí comparing, colocamos aquí curso.getNombre. Y ahora necesitamos esto de aquí.

[04:53] Y estamos implementando automáticamente dentro de nuestro método sort. Entonces Java va a hacer la parte de ordenar la lista. Si queremos utilizar también la clase collections, podemos hacer así: collections.sort. Colocamos aquí nuestra lista de cursos y aquí colocamos comparator.comparing, colocamos aquí curso.getNombre y también va a dar el mismo resultado.

[05:38] Y el .reversed, orden invertido. Listo. Ahora, supongamos que quiero utilizar stream. Por ejemplo vamos a hacer lo siguiente, vamos a colocar aquí cursos.stream().sorted.Comparator, vamos a colocar el comparingInt. ¿Qué va a hacer? Él va a buscar algún método int para hacer la comparación. Aquí encontró getTiempo. Y punto y coma.

[06:29] Utilizamos aquí System.out.println, colocamos cursos. ¿Qué sucede? No ordenó nada. ¿Por qué? Recuerden que en las clases anteriores aquí dije que

esto tiene que retornar una lista nueva. Entonces nosotros necesitamos utilizar el método collect y decir que esto de aquí va a ser una lista. Para eso tenemos que utilizar la interface list, que también está dentro de Java.util.

[07:16] Colocamos aquí curso y aquí colocamos cursoList, el nombre de nuestra variable. Ahora vamos a imprimir. ¿Qué sucedió? Ordenó. ¿Por qué? Por tiempo. ¿Cuál sería el tiempo? Primero, Ruby que es con 10, después está JavaScript con 20, después tenemos PHP con 30 y por último Java con 50.

[08:00] Ahora utilizando lambda. ¿Por qué vamos a utilizar lambda? Porque puede facilitar en ciertas circunstancias, por ejemplo dicen: "No quiero que aparezca Ruby". Entonces, ¿cómo podemos hacer? En Java tenemos que utilizar, tendríamos que utilizar un for simple y ver un if y ver si tiene el Ruby y crear otra nueva lista. Es un poco más complejo.

[08:27] Por eso tenemos aquí la opción utilizando lambda, y está en stream. Escribimos el método filter. En el método filter ponemos por ejemplo curso y aquí abrimos el lambda. ¿En el lambda qué ponemos? Decimos que este curso.getNombre().equalsIgnoreCase vamos a decirle que tiene que ser diferente a Ruby, y vamos a negarlo, porque aquí está diciendo que sí, vamos a negarlo para que sea diferente.

[09:08] Y una vez esto, va a crear una nueva lista sin Ruby. Vamos a ejecutar, y definitivamente, JavaScript, PHP y Java. ¿Qué hizo? Eliminó el Ruby. Tenemos otros métodos para utilizar este int y aprovechar bastante las ventajas que nos dan a partir de Java 8, pero eso lo vamos a ver en las siguientes clases. Esto sería todo por ahora. Muchas gracias y hasta la próxima clase.